المعهد الوطني للبريد والمواصلات
ⵓⵔⴰⵎ ⵓⵏⵟⵓ ⵏ ⵜⵓⵍⵓⵙⵜ ⴰ ⵜⵎⵙⵓⵍⴰⵢ
Institut National des Postes et Télécommunications

agence nationale de réglementation des télécommunications
الوكالة الوطنية لتقنين المواصلات

DATA Engineering

Advanced Learning :

Quiz Generator

powered by OpenAI & Google APIs

Realized By :

MESBAH Abderahmane

Academic Year 2022/2023

Supervised By : Prof. MAHMOUDI Abdelhak

Abstract

This project report introduces an app that converts ChatGPT responses obtained from audio prompts converted to text using Whisper, into engaging Google Form quizzes. It aims to leverage technology to revolutionize how we access and consume information in today's fast-paced digital era. The app provides a convenient means of interacting with technology through audio prompts.

The project's primary goal is to tap into the untapped potential of audio prompts for generating interactive educational content. Through the integration of natural language processing and machine learning, the app streamlines the process of quiz creation, allowing educators, content creators, and learners to transform spoken questions into dynamic and interactive quizzes. By harnessing cutting-edge technologies, the app empowers users to effortlessly generate engaging quizzes from audio prompts, enhancing the accessibility and interactivity of educational content. Its core functionality revolves around using Whisper, a speech to text api that turns any audio prompt into a that can be fed to ChatGPT, an advanced language model, to process text input, extract meaningful responses, and automatically populate a Google Form with quiz questions based on the obtained information. Overall, this app represents a significant advancement in utilizing audio prompts for educational purposes in the digital age.

LINK to the demonstration video : Drive Video.

Contents

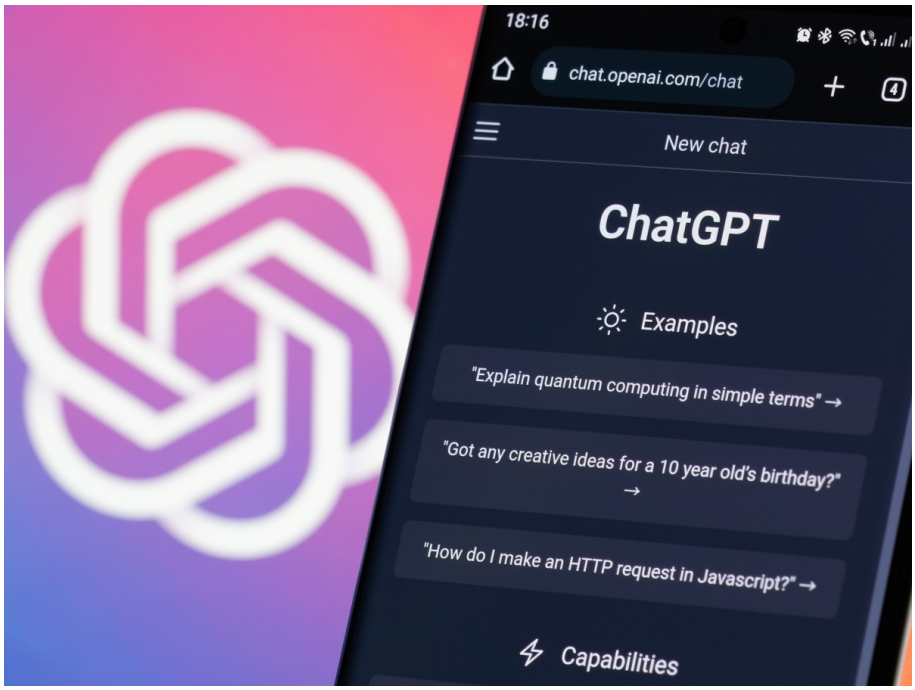
Chapter 1

Implementation Technologies

After a conception which meets the needs of our project, we begin the implementation part of the application we have developed. different tools and APIs were used in the production of this application such as:

1.1 OpenAI Chat API

The OpenAI Chat API is a powerful tool developed by OpenAI that allows developers to integrate natural language processing capabilities into their applications. With the Chat API, developers can create chatbots and conversational agents that can understand and respond to user queries in a conversational manner. It is based on OpenAI's GPT-3.5 language model, which has been trained on a vast corpus of text data and can generate human-like responses.



Using the OpenAI Chat API, developers can build applications that provide personalized and engaging interactions with users. This API has various use

cases, such as virtual assistants, customer support systems, and content generation tools. By leveraging the Chat API's advanced natural language processing capabilities, developers can enhance the user experience and create intelligent conversational interfaces.

1.2 OpenAI Whisper

OpenAI Whisper is an advanced automatic speech recognition (ASR) system developed by OpenAI. It is designed to convert spoken language into written text and offers state-of-the-art performance in speech recognition tasks. Whisper has been trained on a massive amount of multilingual and multitask supervised data, enabling it to accurately transcribe speech in multiple languages and contexts.

Whisper's accurate speech recognition capabilities make it suitable for a wide range of applications. It can be integrated into transcription services, voice assistants, and voice-controlled applications. By leveraging Whisper, developers can enable voice interactions in their applications, making them more accessible and user-friendly. The high accuracy of Whisper's speech recognition can help improve the overall user experience and enable more natural and seamless voice-based interactions.

1.3 Google Forms API

The Google Forms API is a programming interface provided by Google that allows developers to interact with Google Forms programmatically. Google Forms is a web-based tool for creating online surveys and collecting responses. The API enables developers to create, update, and retrieve form templates, as well as submit responses to forms.

By using the Google Forms API, developers can automate the creation and management of Google Forms. This provides the flexibility to integrate Google Forms into custom workflows or applications. Developers can dynamically generate forms, pre-fill form fields with data, and retrieve responses programmatically. The API offers a convenient way to extend the functionality of Google Forms and streamline the process of creating and managing online surveys.

1.4 Streamlit

Streamlit is an open-source Python library that allows developers to create interactive web applications for data science and machine learning projects. It provides a simple and intuitive way to build user interfaces for data exploration, visualization, and model deployment.

With Streamlit, developers can easily convert their Python scripts into interactive apps. It offers a declarative syntax, which means that developers can define the UI elements and their behavior using a straightforward and concise Python code. Streamlit takes care of rendering the UI and automatically updating it as the underlying data or code changes.

Chapter 2

Code explanation

2.1 Importing of the libraries

```
import openai
import re
import whisper
import streamlit as st
from st_custom_components import st_audiorec
import whisper

from apiclient import discovery
from httplib2 import Http
from oauth2client import client, file, tools
```

- `import openai`: This imports the `openai` module, allowing interaction with OpenAI's APIs.
- `import re`: This imports the `re` module for working with regular expressions.
- `import whisper`: This imports the `whisper` module, allowing speech to text transformations.
- `import streamlit as st`: This imports the `streamlit` module and assigns it the alias `st`, simplifying the process of building web applications.
- `from st_custom_components import st_audiorec`: This imports the module from a custom package or module called `st_custom_components`, which provides an audio recording component for Streamlit applications(I found in in GitHub).
- `from apiclient import discovery`: This imports the `discovery` class from the `apiclient` package, which is part of the Google API Client Library.

- `from httplib2 import Http`: This imports the `Http` class from the `httplib2` package, which is used for making HTTP requests.
- `from oauth2client import client, file, tools`: This imports various classes and modules from the `oauth2client` package, which provides authentication and authorization functionality for working with Google APIs.

2.2 Recording code

```
wav_audio_data = st_audiorec()
if wav_audio_data is not None:
    st.audio(wav_audio_data, format='audio/wav')

st.write("While recording, specify the name and number of questions")
filename = st.text_input("Enter the audio filename", "audio.wav")
```

The code snippet records audio using a custom audio recording component. It checks if the recorded audio data exists and, if so, displays it using the st.audio() function. It also writes a message asking the user to specify the name and number of questions while recording. Finally, it allows the user to input a filename for the recorded audio using a text input field.

2.3 Generating the form

```
if st.button("Generate the Quiz"):
```

once the audio file is ready, just click on generate the quiz which launch the following steps

2.3.1 Transcription

```
st.info("Transcription...")

model = whisper.load_model("base")
result = model.transcribe(filename)

st.success(result["text"])
```

The code snippet is triggered when the user clicks a button labeled "Generate the Quiz". When the button is clicked, it displays an information message indicating that the transcription process is underway.

The code then loads a pre-trained Whisper ASR (Automatic Speech Recognition) model called "base". Using this model, it transcribes an audio file specified by the filename variable. The transcribed text is stored in the result variable.

Finally, the transcribed text is displayed to the user as a success message using the st.success() function.

In summary, this code generates a quiz by transcribing an audio file using the Whisper ASR model and provides the transcribed text as feedback to the user.

2.3.2 Quiz generation with ChatCompletion

```
st.info("ChatGPT is generating the quiz...")
openai.api_key = "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"
content = result["text"] + """ create the quiz with multiple choice answers in the same format as following and in the end say finished :
    Title of quiz : "quiz on ..."
Question 1 : ....
    a.
    b.
    c.
    d.
Question 2 : ....
    a.
    b.
    c.
    d.
"""

completion = openai.ChatCompletion.create(model = "gpt-3.5-turbo-0301", messages = [{"role": "user", "content": content }])
requests = completion.choices[0].message.content
```

With this code below we can generate a quiz using ChatGPT. It begins by displaying an information message to inform the user that ChatGPT is in the process of generating the quiz. The code then sets the API key for OpenAI to authenticate API requests.

Next, the code retrieves the transcribed text from the result variable and combines it with a predefined format for creating a quiz. The format includes the title of the quiz and multiple-choice questions with options (a, b, c, d). The concatenated content is then passed as a user message to the ChatGPT model.

The code utilizes OpenAI's ChatCompletion API by creating a chat message with the user role and the prepared content. It sends this message to the gpt-3.5-turbo-0301 model for completion. The response from the API is captured in the completion variable, and the generated quiz requests are extracted from it

2.3.3 Parsing the result using Regular Expression

```
Title = re.findall(r'Quiz on (.*?)\n' , requests)
questions = re.findall(r'Question \d+: (.*?)\n', requests)

a_answers = re.findall(r'a\.(.*?)\n' ,requests)
b_answers = re.findall(r'b\.(.*?)\n' ,requests)
c_answers = re.findall(r'c\.(.*?)\n' ,requests)
d_answers = re.findall(r'd\.(.*?)\n' ,requests)
```

this code segment employs regular expressions to parse the generated quiz requests and extract the title, questions, and multiple-choice answer options in order to be fed to google forms API

2.3.4 Creating the body of the form

```
q = []

for i in range(len(questions)):
    q.append({
        "createItem": {
            "item": {
                "title": questions[i],
                "questionItem": {
                    "question": {
                        "required": True,
                        "choiceQuestion": {
                            "type": "RADIO",
                            "options": [
                                {"value": a_answers[i]},
                                {"value": b_answers[i]},
                                {"value": c_answers[i]},
                                {"value": d_answers[i]}
                            ],
                            "shuffle": True
                        }
                    }
                },
            },
            "location": {
                "index": i
            }
        }
    })
```

This code segment constructs a list of dictionaries representing the quiz questions and their answer options, following a specific format.

2.3.5 The setup for google forms api

```
SCOPES = "https://www.googleapis.com/auth/forms.body"
DISCOVERY_DOC = "https://forms.googleapis.com/$discovery/rest?version=v1"

st.info("Wait for authentication...")

store = file.Storage('token.json')
creds = None
if not creds or creds.invalid:
    flow = client.flow_from_clientsecrets('key.json', SCOPES)
    creds = tools.run_flow(flow, store)

form_service = discovery.build('forms', 'v1', http=creds.authorize(
    Http()), discoveryServiceUrl=DISCOVERY_DOC, static_discovery=False)

# Request body for creating a form
NEW_FORM = {
    "info": {
        "title": Title[0],
    }
}

# Request body to add a multiple-choice question
NEW_QUESTION = {
    "requests": q}
```

This code segment sets up the necessary authentication and builds the Google Forms service. It defines the request bodies for creating a form and adding multiple-choice questions, based on the extracted title and the list of dictionaries representing the questions and answer options.

2.3.6 Authentication and retrieving the form

```
# Creates the initial form
result = form_service.forms().create(body=NEW_FORM).execute()

# Adds the question to the form
question_setting = form_service.forms().batchUpdate(formId=result["formId"], body=NEW_QUESTION).execute()

# Prints the result to show the question has been added
get_result = form_service.forms().get(formId=result["formId"]).execute()
st.success("Link to the form:")
st.info(get_result['responderUri'])
```

Using the code below we create a Google Form by making a request to the Google Forms API. According to the documentation, It uses the NEW_FORM dictionary as the request body and retrieves the result, including the form ID. The code then adds a question to the form by making a batch update request, specifying the form ID and using the NEW_QUESTION dictionary as

the request body. The updated form is retrieved using a get request, and the responder link to access the form is displayed to the user.

Conclusion

This project was about creating a quiz generator app using different AI tools and APIs, I did a little analysis of user situations. I have also provided the architecture of the application, including the interaction workflow of the user with the application and details of our model in this report and in the video.

This project has allowed me to gain personal and professional experience. It was beneficial to me because I had the chance to improve my knowledge in the world of AI and my capacities in problem solving. what a formidable project to end this year.

Bibliographie

https://github.com/stefanrmmr/streamlit_audio_recorder
https://github.com/openai/whisper
https://platform.openai.com/docs/introduction
https://developers.google.com/forms/api/quickstart/python?hl=fr
https://docs.streamlit.io/
The project Github Repository
https://github.com/ABMesbh/Quiz_generator_using_OpenAI-Goog
APIs